AF/2154
120

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:
Sai V. Allavarpu
Rajeev Angal

Serial No. 09/556,069

Filed: April 21, 2000

For: Generic and Dynamic Mapping of Abstract Syntax Notation (ASN1) to and from Interface Definition Language for Network Management

§ Group Art Unit: 2154
§
§ Examiner: Lin, Wen Tai
§
§ Atty. Dkt. No.: 5181-46100
§ P4460

CERTIFICATE OF MAILING
37 C.F.R. § 1.8

I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as First Class Mail in an envelope addressed to Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date indicated below:

Robert C. Kowert
Name of Registered Representative

June 28, 2004
Date

Signature

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir/Madam:

Further to the Notice of Appeal filed April 26, 2004, Appellants present this Appeal Brief. Appellants respectfully request that this appeal be considered by the Board of Patent Appeals and Interferences.

I. REAL PARTY IN INTEREST

The subject application is owned by Sun Microsystems, Inc., a corporation organized and existing under and by virtue of the laws of the State of Delaware, and having its principal place of business at 4150 Network Circle, Santa Clara, CA 95054, as evidenced by the assignment recorded at Reel 010993, Frame 0846.

II. RELATED APPEALS AND INTERFERENCES

No other appeals or interferences are known which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1 – 34 are pending in the present application and are the subject of this appeal. Claims 1, 10 – 11, 13, 22 – 23, 25, 28 – 29, 31 and 33 stand finally rejected under 35 U.S.C. § 102(e). Claims 2 – 9, 12, 14 – 21, 24, 26 – 27, 30, 32 and 34 stand finally rejected under 35 U.S.C. § 103(a). A copy of claims 1 – 34, as on appeal (incorporating all amendments), is included in the Appendix hereto.

IV. STATUS OF AMENDMEMNTS

No amendment to the claims has been filed subsequent to the final rejection. The Appendix hereto reflects the current state of the claims.

V. SUMMARY OF THE INVENTION

A middleware standard used extensively in network management is the Common Object Request Broker Architecture (CORBA), which is provided by the Object Management Group (OMG). CORBA specifies a system that provides interoperability between objects in a heterogeneous, distributed environment and in a way transparent to the programmer. Its design is based on the OMG Object Model, which defines common

object semantics for specifying the externally visible characteristics of objects in a standard and implementation-independent way. In this model, clients request services from objects (which will also be called servers) through a well-defined interface. This interface is specified in the OMG Interface Definition Language (IDL). In CORBA, a client accesses an object by issuing a request to the object. The request is an event, and it carries information including an operation and actual parameters, if any. A central component of CORBA is the Object Request Broker (ORB). The ORB encompasses the communication infrastructure necessary to identify and locate The ORB acts as the middleware between clients and servers. In the CORBA model, a client can request a service without knowing anything about what servers are attached to the network. The various ORBs receive the requests, forward them to the appropriate servers, and then hand the results back to the client. See page 5, lines 6-30.

The Object Management Group (OMG) and Joint Inter-Domain Management (JIDM) have defined Interface Definition Language (IDL) for network management, which is used to access object instance data and may be used across a plurality of programming languages and across a plurality of platforms. IDL allows programmers to write only one set of interfaces for a particular object across multiple programming languages, rather than having to write a new set of interfaces for each programming language. Although the use of IDL as a programming-language-independent and platform-independent interface to define CORBA services and objects greatly improves the ease with which network objects may be managed, in the prior art an IDL interface must be defined and compiled for each TMN object to be managed. *See* page 7, lines 8-27.

As opposed to the prior art solution of defining and compiling a separate IDL interface for each managed object, embodiments of the present invention provide for a system and method for generic and dynamic mapping of managed object metadata by accessing a single generic converter interface for converting or mapping data types in

communications to managed objects such that the converting is generic to the managed objects. *See* page 8, lines 14-25.

In one embodiment, CORBA-based Telecommunications Management Network (TMN) manager applications may be communicatively coupled to a CORBA Object Request Broker (ORB). The manager applications may be operable to send Interface Definition Language (IDL) requests and receive IDL responses and CORBA events through the CORBA ORB. A CORBA gateway may also be communicatively coupled to the CORBA ORB and be operable to communicate with the CORBA ORB via communications methods such as the Internet Inter-Object Protocol (IIOP) and IDL. *See* Fig. 2, and page 14, line 27 – page 15, line 4.

An enterprise manager may be coupled to the CORBA gateway via a proprietary or platform-dependent interface such as Portable Management Interface (PMI) from Sun Microsystems, Inc. The enterprise manager may include various enterprise management components such as a Management Information System (MIS), etc. Also coupled to the enterprise manager via PMI may be one or more PMI applications. In one embodiment, the CORBA gateway may translate the CORBA manager requests from IDL to PMI requests. Similarly, the CORBA gateway may translate the enterprise manager PMI responses and PMI events to IDL/IIOP responses and events which may be passed on through the CORBA ORB to the manager applications in the form of IDL responses and CORBA events. *See* Fig. 2, and page 15, lines 9 – 24.

In one embodiment, manager applications may request information regarding managed objects on the network via the CORBA Gateway. The managed objects may represent devices such as cell phones, cell phone towers, phone systems, faxes, routers, switches, etc., which may be interconnected via networks. The data related to such objects may exist in a variety of formats, and so must be translated to forms readily understood by the manager applications. This function may be performed by a component of the CORBA Gateway. More specifically, a mapping system may be

invoked by the CORBA Gateway to translate managed object data and metadata between various data formats, such as OMG IDL and ASN1. *See* page 15, line 26 – page 16, line 24.

In one embodiment, a manager application is communicatively coupled to a converter framework via an interface definition language such as IDL. The converter framework is coupled to various converter modules from a set of converter implementations. The converter framework is further coupled to the enterprise manager through an abstract syntax notation or other metadata notation language, such as ASN1. *See* Fig. 5, and page 22, lines 17 – 24.

In one embodiment, the converter modules are plug-in modules which are operable to plug into the converter framework. The converter modules may then provide various functional mappings of managed object metadata between the interface definition language and the abstract syntax notation as invoked by the converter framework. In one embodiment, each converter module may be operable to map managed object metadata types between IDL and ASN1, according to that module's particular mapping. In an alternate embodiment, a converter module may map managed object metadata types between IDL and a transaction language, such as Transaction Language One (TL-1). The combination of using IDL and a generic type <any> provides an efficient, generic solution to mapping data types across multiple platforms, multiple programming languages, and multiple object classes. The fact that the framework may accommodate a variety of plug-in modules for mapping other data types to and from IDL further enhances the suitability of the system and method for a generic and expandable solution to data and metadata type conversions. *See* page 22, line 26 – page 23, line 12.

VI. ISSUES

1. Whether claims 1, 10 – 11, 13, 22 – 23, 25, 28 – 29, 31 and 33 are anticipated by Carre (U.S. Patent 6,282,579) under 35 U.S.C. § 102(e).

2. Whether claims 2 – 6, 12, 14 – 18, 24, and 26 – 27 are patentable under 35 U.S.C. § 103(a) over Carre (U.S. Patent 6,282,579) in view of Applicant Admitted Prior Art (AAPA)

3. Whether claims 7 – 9, 19 – 21, 32 and 34 are patentable under 35 U.S.C. § 103(a) over Carre (U.S. Patent 6,282,579).

4. Whether claim 30 is patentable under 35 U.S.C. § 103(a) over Carre (U.S. Patent 6,282,579) in view of Goldberg et al. (U.S. Patent 6,496,833hereinafter "Goldberg")

VII. GROUPING OF CLAIMS

Claims 1 – 8, 10 – 20, and 22 – 34 stand or fall together.

Claims 9 and 21 stand or fall together.

The above claim groupings are for purposes of this appeal only. The reasons why each group of claims is believed to be separately patentable are explained below in the Argument.

VIII. ARGUMENT

A. Claims 1 – 8, 10 – 10, and 22 – 34

The Examiner rejected claims 1, 10 – 11, 13, 22 – 23, 25, 28 – 29, 31 and 33 under 35 U.S.C. § 102(e) as being anticipated by Carre (U.S. Patent 6,282,579) (hereinafter "Carre"). Appellants respectfully traverse these rejections in light of the following remarks.

Contrary to the Examiner's assertion, Carre does not teach "receiving a plurality of communications each pertaining to a different one of a plurality of managed objects from a management server for the managed objects, wherein each communication comprises data typed according to an abstract syntax notation, and accessing a converter interface for each communication for converting the abstract syntax notation data types of each communication to interface definition language data types, wherein the same converter interface is accessed for each of the managed objects such that the converting is generic to the managed objects," as recited in Appellants' claim 1.

Contrary to the Examiner's assertions, Carre appears to provide a specific IDL interface, which is defined and compiled for each managed object, as in the prior art described in the Description of the Relevant Art section of the Appellants' specification. In the prior art, as discussed at p. 7 lines 14-27 of the Appellants' specification, CORBA communications are provided for by specific IDL interfaces defined and compiled for each managed object

On p. 8 of the Final Action, the Examiner refers to col. 5, lines 9-20 of Carre as teaching that the agent A consists of one or more OSI objects OA and to col. 5, lines 30-39 of Carre as teaching that the OA objects are converted from ASN.1 to IDL. From these two references, the Examiner concludes that Carre's converter "must be generic enough to handle the plurality of OA and OM objects." Appellants assert that this

conclusion is unfounded in view of Carre's teachings. Although Carre does teach that "component A consists of one or more OSI objects OA" (col. 5, lines 12-13), Carre also clearly teaches that the "communication entity GDMO/C++ consists of one or more *specific* access objects" (col. 5, lines 21-22). Carre further describes how objects not specified in CORBA "interact with one another and with the objects CO and SO via *specific* interface units" (col. 4, lines 22-27). Thus, Carre includes a *specific* access object for each OSI object (like the prior art described in Appellants' Description of the Relevant Art section). Carre also refers to a specified interface for *the* [singular] object OA (col. 5, lines 30-31). Hence, nothing in Carre teaches, or even suggests, a converter that is generic to a plurality of different managed objects. Thus, the Examiner's statement that Carre's converter must be generic amounts to nothing more than hindsight speculation by the Examiner.

Anticipation requires the presence in a single prior art reference disclosure of each and every element of the claimed invention, arranged as in the claim. *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 221 USPQ 481, 485 (Fed. Cir. 1984). The identical invention must be shown in as complete detail as is contained in the claims. *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). Carre clearly does not anticipate Appellants' claimed invention.

Since the rejection is not supported by the teaching of the cited reference, Appellants respectfully request reversal of the Examiner's rejection of claim 1, along with its dependent claims 2 – 12. Independent claims 13, 25, and 31, along with their respective dependent claims, are also believed patentably for at least the above reason.

B. Claims 9 and 21

The Examiner rejected claims 9 and 21 under 35 U.S.C. § 103(a) as being unpatentable over Carre. Appellants respectfully traverse this rejection in light of the following remarks.

Appellants can find no teaching or suggestion in Carre of a method "wherein the first data type comprises an object data type in the abstract syntax notation, wherein the second data type comprises a choice structure of the interface definition language, and wherein the choice structure comprises: a generic value field; a plurality of data types; and a selector, wherein the selector is an index whose value determines which of the plurality of data types is used to interpret the value in the generic value field," as recited in Appellants' claim 9.

Specifically, Appellants can find no reference or suggestion in Carre regarding a choice structure that includes a plurality of data types and a selector, wherein the selector is an index whose value determines which of the plurality of data types is used to interpret the value in the generic value field. In contrast, Carre teaches a conversion of an first address type to a correspondent, second address type that "contains as an additional value the address value according to the other addressing mode [of first address type]." Carre explains further, "both address values are transported in the second type and are available both during the transport of a message with this type and at the target object." (col. 2, lines 30-42, see also col.. 2, lines 7-19, col. 6, lines 1-21). In other words, Carre teaches supplying *multiple values* for a *single converted address type.*

The Examiner admits on page 6 of the Final Action, that Carre does not specifically teach a choice structure comprising a plurality of data types and a selector, wherein the selector is an index whose value determines which of the plurality of data types is used to interpret the value in the generic value field, but that since "these corresponding pairs are known data types for describing OSI and CORBA objects in their respective languages, it is obvious that, under Carre's methodology (as described in Fig. 2a and col. 5, lines 24 – 39), a mapping between these pairs must be established, because Carre's CMISE services require type interoperability between the data types expressed in the abstract syntax notation and the interface definition language." Appellants assert, however, that such a choice structure is not a well-known data type for describing OSI

and CORBA objects. Carre clearly does not describe such a structure. Carre does teach that type interoperability is necessary (col. 5, lines 30-33). However, Carre teaches that such interoperability is obtained through conversions of a data type to a structure including a single data type and two values, one pre-conversion and one post-conversion. Thus, Carre is clearly teaching away from Appellants' use of a structure with a single value, multiple data types, and a selector determining which data type is used to interpret the value.

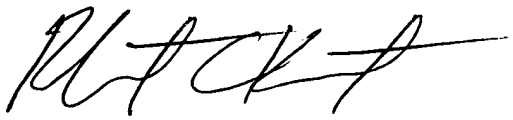
Accordingly, claim 9 is believed patentably distinct over the cited art for at least this further reason. Claim 21 recites similar limitations, and is likewise believed patentably distinct for at least this further reason.

IX. CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejections of claims 1 – 34 were erroneous, and reversal of his decision is respectfully requested.

The Commissioner is authorized to charge the appeal brief fee of $330.00 and any other fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-46100/RCK. This Appeal Brief is submitted in triplicate along with a return receipt postcard.

Respectfully submitted,



Robert C. Kowert
Reg. No. 39,255
Attorney for Appellants

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: June 28, 2004

X. APPENDIX

The claims on appeal are as follows.

1. A method for mapping managed object metadata, the method comprising:

receiving a plurality of communications each pertaining to a different one of a plurality of managed objects from a management server for the managed objects, wherein each communication comprises data typed according to an abstract syntax notation; and

accessing a converter interface for each communication for converting the abstract syntax notation data types of each communication to interface definition language data types, wherein the same converter interface is accessed for each of the managed objects such that said converting is generic to the managed objects, wherein said converting comprises:

inputting a first data type from a first set of data types, wherein the first set of data types is expressed in the abstract syntax notation, and wherein the abstract syntax notation comprises a language for describing data;

determining a corresponding second data type from a second set of data types, wherein the second set of data types is expressed in the interface definition language, wherein the interface definition language comprises a language for implementing interfaces to managed objects, wherein the interface definition language is operable across a plurality of platforms and across a plurality of programming languages, and wherein the interface definition language is class independent; and

returning the second data type.

2. The method of claim 1, wherein the managed object metadata comprises metadata concerning a telephone system.

3. The method of claim 1, wherein the managed object metadata comprises metadata concerning a network switch.

4. The method of claim 1, wherein the metadata comprises type information about an attribute of one of the managed objects.

5. The method of claim 1, wherein the metadata comprises type information about an action of one of the managed objects.

6. The method of claim 1, wherein the metadata comprises type information about a notification of one of the managed objects.

7. The method of claim 1, wherein the first data type comprises a primitive data type in the abstract syntax notation, and wherein the second data type comprises a generic primitive data type in the interface definition language.

8. The method of claim 1, wherein the first data type comprises an object data type in the abstract syntax notation, and wherein the second data type comprises a sequence of a generic primitive data type in the interface definition language.

9. The method of claim 1, wherein the first data type comprises an object data type in the abstract syntax notation, wherein the second data type comprises a choice structure of the interface definition language, and wherein the choice structure comprises:

a generic value field;

a plurality of data types; and

a selector, wherein the selector is an index whose value determines which of the plurality of data types is used to interpret the value in the generic value field.

10. The method of claim 1, wherein the abstract syntax notation comprises Abstract Syntax Notation One.

11. The method of claim 1, wherein the abstract syntax notation comprises Guidelines for Managed Objects (GDMO).

12. The method of claim 1, wherein the interface definition language is operable to provide a mapping which is applicable to any managed object class.

13. A method for mapping managed object metadata, the method comprising:

receiving a plurality of communications each pertaining to a different one of a plurality of managed objects from a manager for the managed objects, wherein each communication comprises data typed according to an interface definition language;

accessing a converter interface for each communication for converting the interface definition language data types of each communication to abstract syntax notation data types, wherein the same converter interface is accessed for each of the managed objects such that said converting is generic to the managed objects, wherein said converting comprises:

inputting a first data type from a first set of data types, wherein the first set of data types is expressed in the interface definition language, wherein the

interface definition language comprises a language for implementing interfaces to managed objects, wherein the interface definition language is operable across a plurality of platforms and across a plurality of programming languages, and wherein the interface definition language is operable to provide a mapping which is applicable to any managed object class;

determining a corresponding second data type from a second set of data types, wherein the second set of data types is expressed in ~~an~~ the abstract syntax notation, and wherein the abstract syntax notation comprises a language for describing data; and

returning the second data type.

14. The method of claim 13, wherein the managed object metadata comprises metadata concerning a telephone system.

15. The method of claim 13, wherein the managed object metadata comprises metadata concerning a network switch.

16. The method of claim 13, wherein the metadata comprises type information about an attribute of one of the managed objects.

17. The method of claim 13, wherein the metadata comprises type information about an action of one of the managed objects.

18. The method of claim 13, wherein the metadata comprises type information about a notification of one of the managed objects.

19. The method of claim 13, wherein the first data type comprises a generic primitive data type in the interface definition language, and wherein the second data type comprises a primitive data type in the abstract syntax notation.

20. The method of claim 13, wherein the first data type comprises a sequence of a generic primitive data type in the interface definition language, and wherein the second data type comprises an object data type in the abstract syntax notation.

21. The method of claim 13, wherein the first data type comprises a choice structure of the interface definition language, wherein the choice structure comprises:

a generic value field;

a plurality of data types; and

a selector, wherein the selector is an index whose value determines which of the plurality of data types is used to interpret the value in the generic value field; and

wherein the second data type comprises an object data type in the abstract syntax notation.

22. The method of claim 13, wherein the abstract syntax notation comprises Abstract Syntax Notation One.

23. The method of claim 13, wherein the abstract syntax notation comprises Guidelines for Managed Objects (GDMO).

24. The method of claim 13, wherein the interface definition language is operable to provide a mapping which is applicable to any managed object class.

25. A computer system for mapping managed object metadata, wherein the system comprises:

a CPU;

a memory coupled to the CPU, wherein the memory stores program instructions executable by the CPU, and wherein the program instructions are executable to implement a generic converter for converting abstract syntax notation data types to interface definition language data types for metadata pertaining to a plurality of different managed objects, wherein the same converter interface is accessed for each of the managed objects, wherein the generic converter is configured to:

input a first data type from a first set of data types, wherein the first set of data types is expressed in the abstract syntax notation, and wherein the abstract syntax notation comprises a language for describing object data;

determine a corresponding second data type from a second set of data types, wherein the second set of data types is expressed in the interface definition language, wherein the interface definition language comprises a language for implementing interfaces to managed objects, wherein the interface definition language is operable across a plurality of platforms and across a plurality of programming languages, and wherein the data types in the interface definition language are class independent; and

return the second data type.

26. The computer system of claim 25, wherein the managed object metadata comprises metadata concerning a telephone system.

27. The computer system of claim 25, wherein the managed object metadata comprises metadata concerning a network switch.

28. The computer system of claim 25, wherein the abstract syntax notation comprises Abstract Syntax Notation One.

29. The computer system of claim 25, wherein the abstract syntax notation comprises Guidelines for Managed Objects (GDMO).

30. The computer system of claim 25, wherein in determining the corresponding second data type from the second set of data types, the program instructions are executable to look up the second data type in one or more lookup tables.

31. A carrier medium comprising program instructions for mapping managed object metadata, wherein the program instructions are computer-executable to implement:

receiving a plurality of communications each pertaining to a different one of a plurality of managed objects from a management server for the managed objects, wherein each communication comprises data typed according to an abstract syntax notation;

accessing a converter interface for each communication for converting the abstract syntax notation data types of each communication to interface definition language data types, wherein the same converter interface is accessed for each of the managed objects such that said converting is generic to the managed objects, wherein said converting comprises:

inputting a first data type from a first set of data types, wherein the first set of data types is expressed in the abstract syntax notation, and wherein the abstract syntax notation comprises a language for describing data;

determining a corresponding second data type from a second set of data types, wherein the second set of data types is expressed in the interface definition language, wherein the interface definition language comprises a language for implementing interfaces to managed objects, wherein the interface definition language is operable across a plurality of platforms and across a plurality of programming languages, and wherein the interface definition language is class independent; and

returning the second data type.

32. The carrier medium of claim 31, wherein the first data type comprises an object data type in the abstract syntax notation, and wherein the second data type comprises a sequence of a generic primitive data type in the interface definition language.

33. A carrier medium comprising program instructions for mapping managed object metadata, wherein the program instructions are computer-executable to implement:

receiving a plurality of communications each pertaining to a different one of a plurality of managed objects from a manager for the managed objects, wherein each communication comprises data typed according to an interface definition language;

accessing a converter interface for each communication for converting the interface definition language data types of each communication to abstract syntax notation data types, wherein the same converter interface is

accessed for each of the managed objects such that said converting is generic to the managed objects, wherein said converting comprises:

inputting a first data type from a first set of data types, wherein the first set of data types is expressed in the interface definition language, wherein the interface definition language comprises a language for implementing interfaces to managed objects, wherein the interface definition language is operable across a plurality of platforms and across a plurality of programming languages, and wherein the interface definition language is operable to provide a mapping which is applicable to any managed object class;

determining a corresponding second data type from a second set of data types, wherein the second set of data types is expressed in ~~an~~ the abstract syntax notation, and wherein the abstract syntax notation comprises a language for describing data; and

returning the second data type.

34. The carrier medium of claim 33, wherein the first data type comprises a sequence of a generic primitive data type in the interface definition language, and wherein the second data type comprises an object data type in the abstract syntax notation.